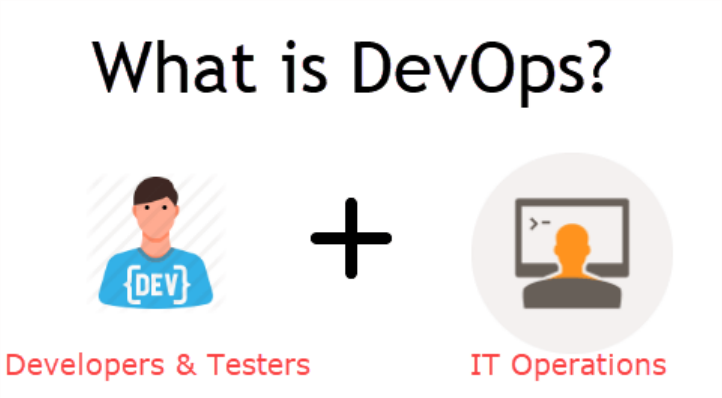
DevOps

What is DevOps

- DevOps is a culture which promotes collaboration between Development and Operations Team to deploy code to production faster in an automated & repeatable way.
- The word 'DevOps' is a combination of two words 'development' and 'operations.'



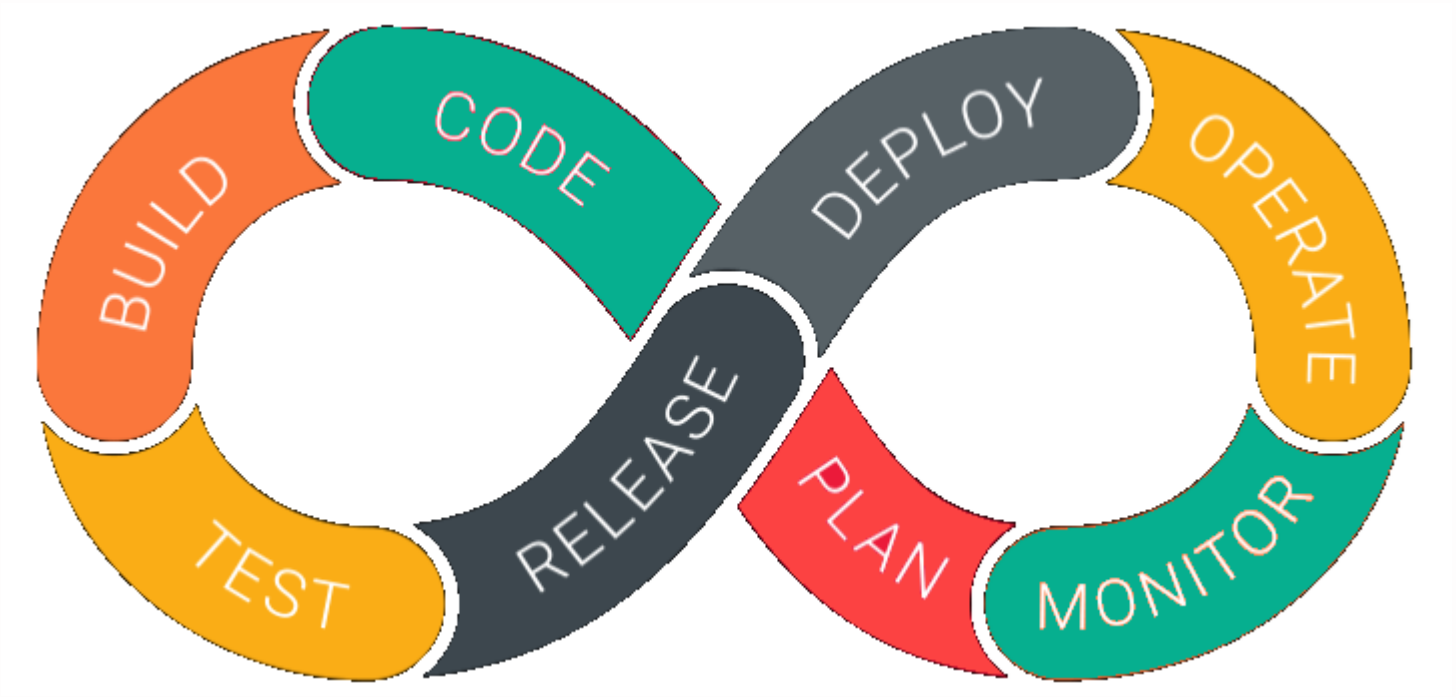
Why Devops?

- Before DevOps:-
  - The development and operation team worked in complete isolation
  - Testing and Deployment were isolated activities done after design-build. Hence they consumed more time than actual build cycles
  - Team members are spending a large amount of their time in testing, deploying, and designing instead of building the project
  - Manual code deployment leads to human errors in production
  - Coding & operation teams have their separate timelines and are not in synch causing further delays

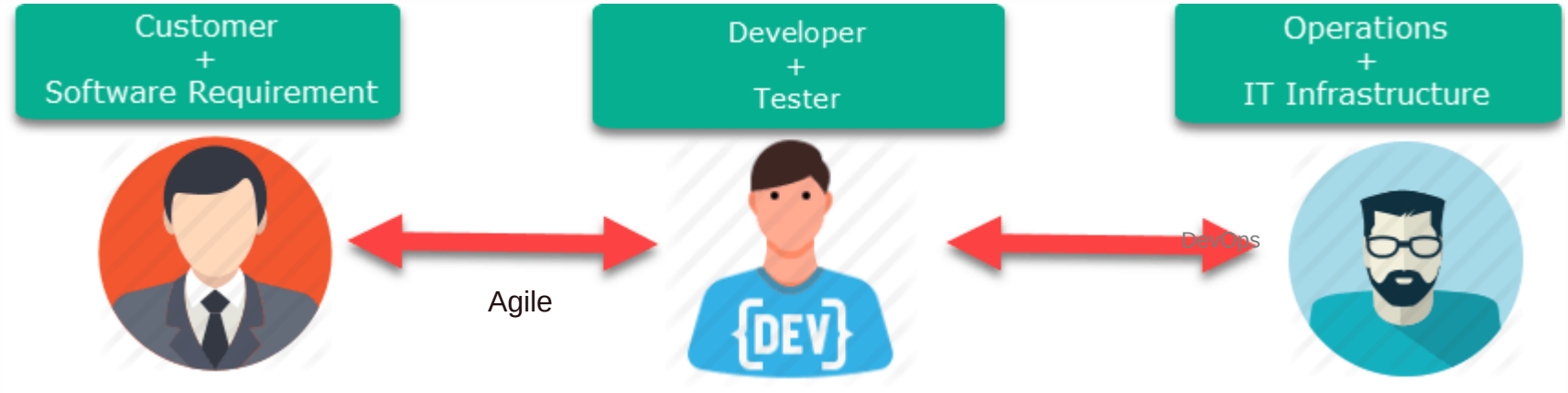
Why DevOps?

- Predictability – Lower failure rates for new Releases
- Reproducability – Earlier versions can be restored anytime
- Maintainability -
- Time to Market – streamlined software delivery
- Quality -
- Reduced Risk - Reduce defects
- Resiliency – Changes are Auditable
- Cost Efficiency -
- Breaks Larger code base into small chunks – follows Agile methodology

DevOps Lifecycle



DevOps vs Agile



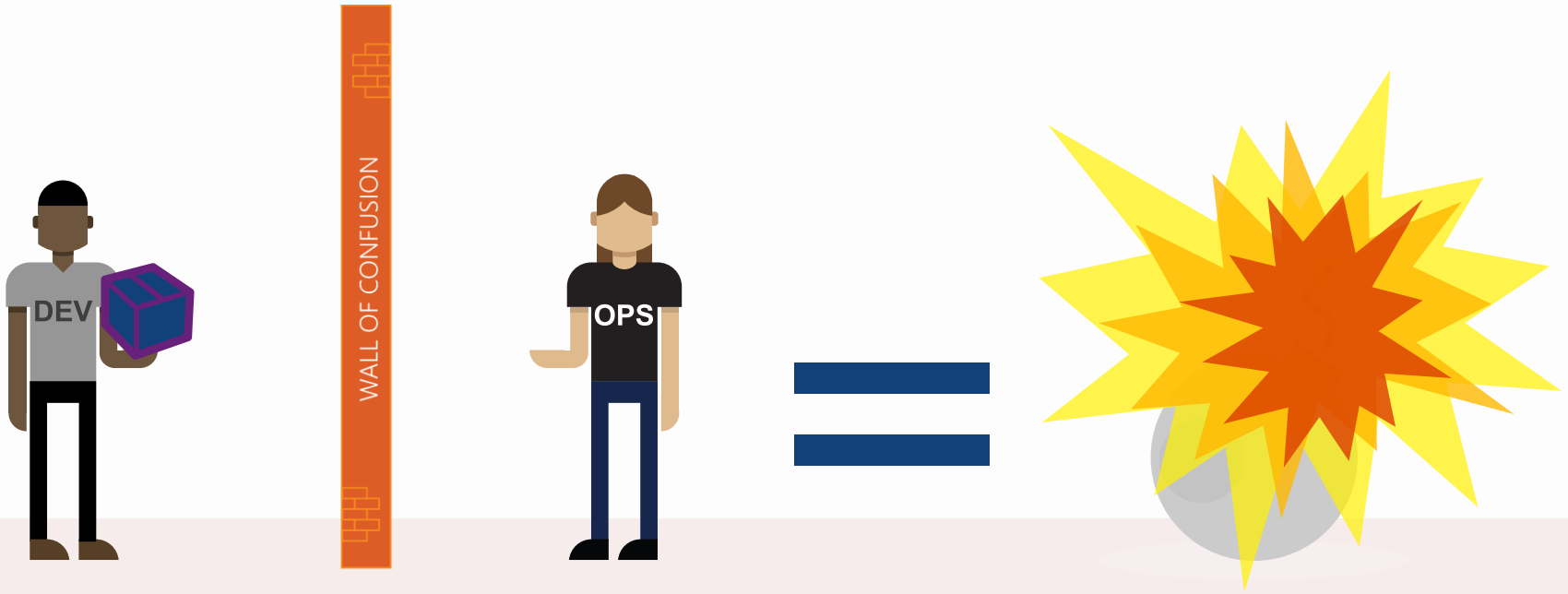
DevOps Principles

- Customer-Centric Action
- End-To-End Responsibility
- Continuous Improvement
- Automate Everything
- Work as one Team
- Monitor and Test Everything

DevOps Automation Tools

- Different categories of Automation:
  - Infrastructure Automation
    - AWS, Azure
  - Configuration Management
    - Chef, Puppet
  - Deployment Automation
    - Jenkins
  - Performance Management
    - App Dynamic
  - Log Management
    - Splunk
  - Monitoring
    - Nagios

Traditional Development and Operations



"**DevOps** is development and operations collaboration"

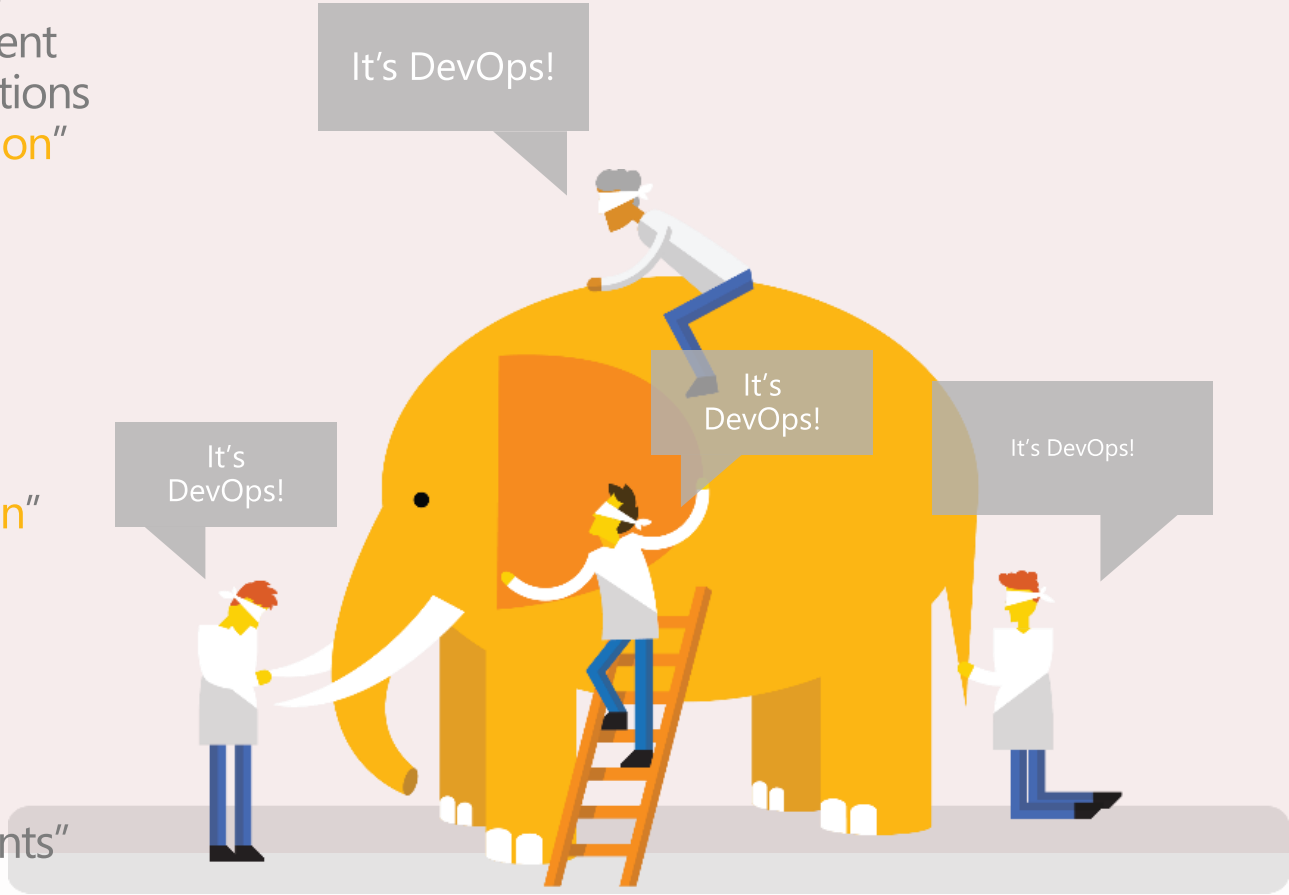
"**DevOps** is treating your infrastructure as code"

"**DevOps** is using automation"

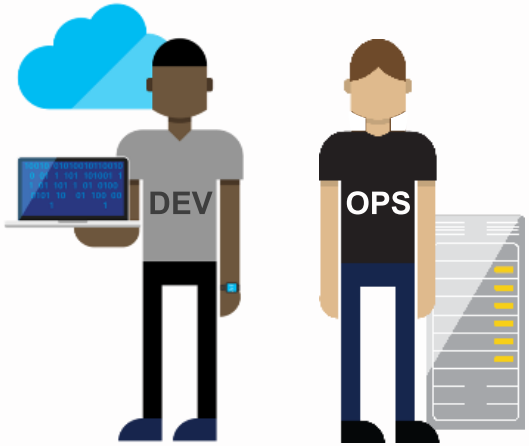
"**DevOps** is feature switches"

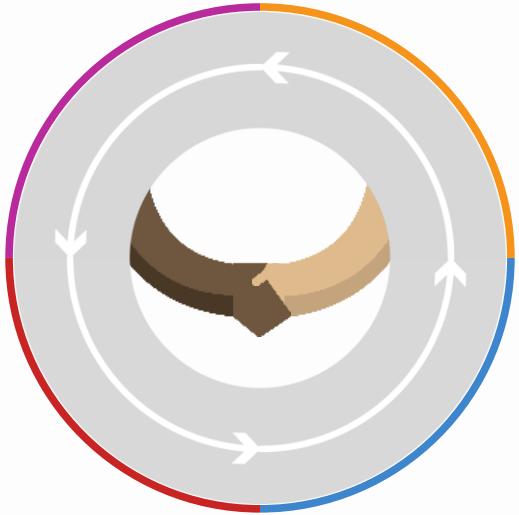
"**DevOps** is small deployments"

"Kanban for Ops?"



DevOps: the three stage conversation

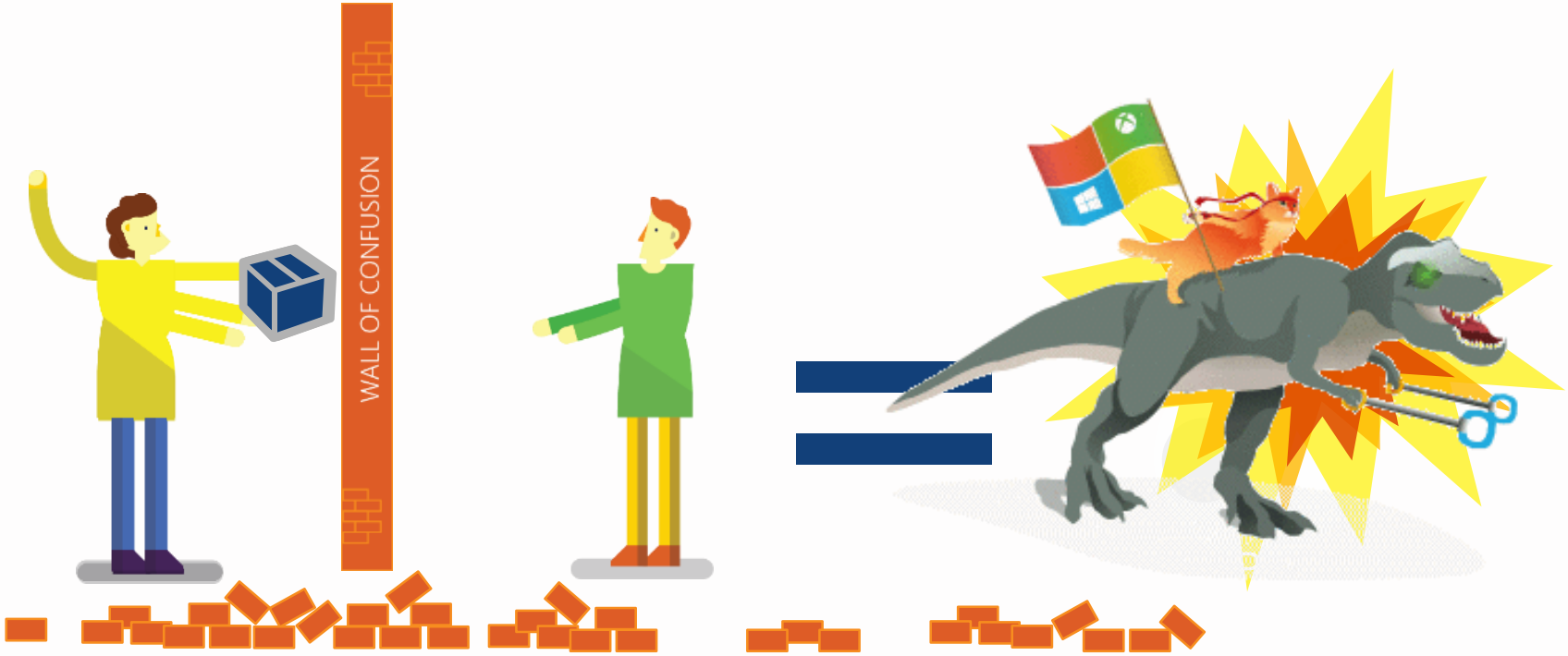




1	People	2	Process	3	Products

HABITS FOR DEVOPS SUCCESS

Happy DevOps



Git

What is Git?

- *Git is a Distributed version Control System.*
- *Directory Control Management System.*
- *Tree History Storage System.*

Why use Source Control Systems ?

- What is ?

- SCS are a tool that helps keeping versions of the code
- SCS allow multiple developers to work on the same code with minimum amount of collisions

- Why use ?

- Keeps the developing process simple
- All files are hosted (Github)

What is GitHub?

- GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

- GitHub essentials like *repositories*, *branches*, *commits*, and *Pull Requests*.

- **No coding necessary for starting or learning github.**

Some basic Terminology

- git = the shell command to work with Git
- repo = Repository, where the code for a given project is kept
- commit = verb, means push the code to the server (in Git, commit = (commit + push)

Branching

- Branching is the way to work on different versions of a repository at one time.
- By default your repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.
- When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

Git

What is Git?

- *Git is a Distributed version Control System.*
- *Directory Control Management System.*
- *Tree History Storage System.*

Why use Source Control Systems ?

- What is ?

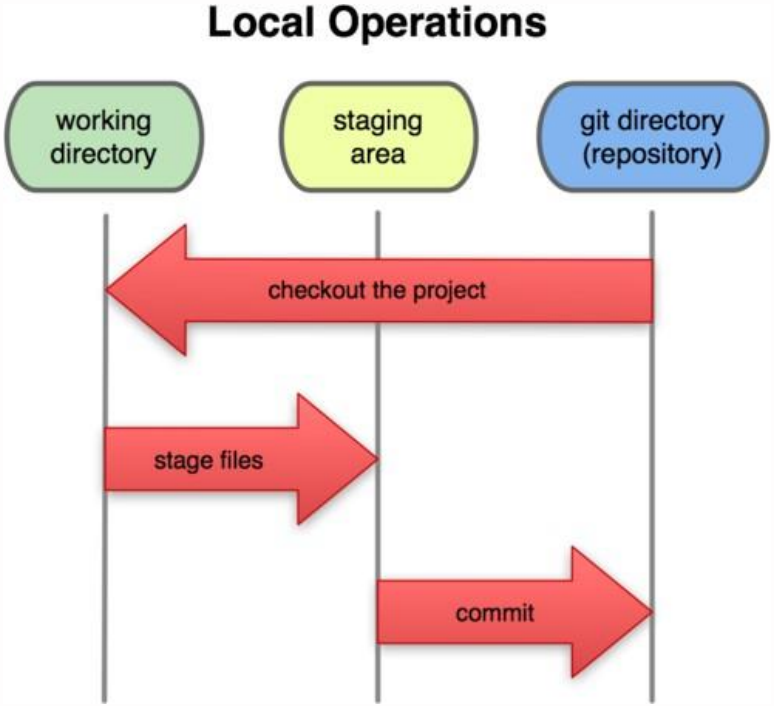
- SCS are a tool that helps keeping versions of the code
- SCS allow multiple developers to work on the same code with minimum amount of collisions

- Why use ?

- Keeps the developing process simple
- All files are hosted (Github)

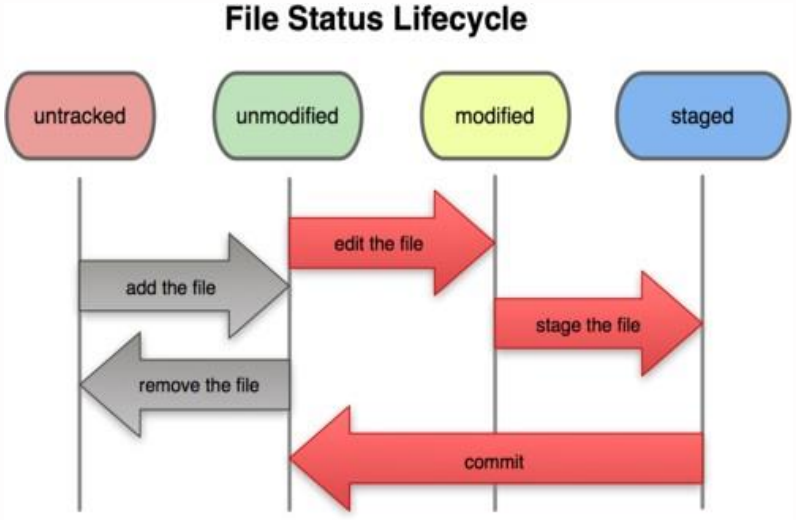
Basic working of Git

- In your local copy on git, files can be:
  - In your local repo
    - (committed)
  - Checked out and modified, but not yet committed
    - (working copy)
  - Or, in-between, in a "staging" area
    - Staged files are ready to be committed.
- A commit saves a snapshot of all staged state.



Cont'd…

- **Modify** files in your working directory.
- **Stage** files, adding snapshots of them to your staging area.
- **Commit**, which takes the files in the staging area and stores that snapshot permanently to your Git directory.



What is GitHub?

- GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

- GitHub essentials like *repositories*, *branches*, *commits*, and *Pull Requests*.

- **No coding necessary for starting or learning github.**

Some basic Terminology

- git = the shell command to work with Git
- repo = Repository, where the code for a given project is kept
- commit = verb, means push the code to the server (in Git, commit = (commit + push)

Branching

- Branching is the way to work on different versions of a repository at one time.
- By default your repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.
- When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

LTI
Let's Solve

Celebrating 20 Years

Maven

Larsen & Toubro
Group Company

Maven

- Maven, - *accumulator of knowledge,*
- A standard way to build the projects, a clear definition of what the project consisted of, an easy way to publish project information and a way to share JARs across several projects.
- Maven is build tool
  - to build deployable artifacts from source code.
  - preprocessing, compilation, packaging, testing, and distribution
- Project Management Tool
  - To Help In Project Management
  - run reports, generate a web site, and facilitate communication among members of a working team.

Maven- Convention over Configuration

- Maven is based on conventions.
- The Location of source code is known is because of the convention used by It.
- Without customization, source code is assumed to be in ${basedir}/src/main/java
- Having a source in the correct directory, is based requirement if that is done Maven will take care of the rest.

Maven vs ANT

- Apache Ant
  - No formal conventions like a common project directory structure or default behavior.
  - It is procedural. Need to tell Ant exactly what to do and when to do it.

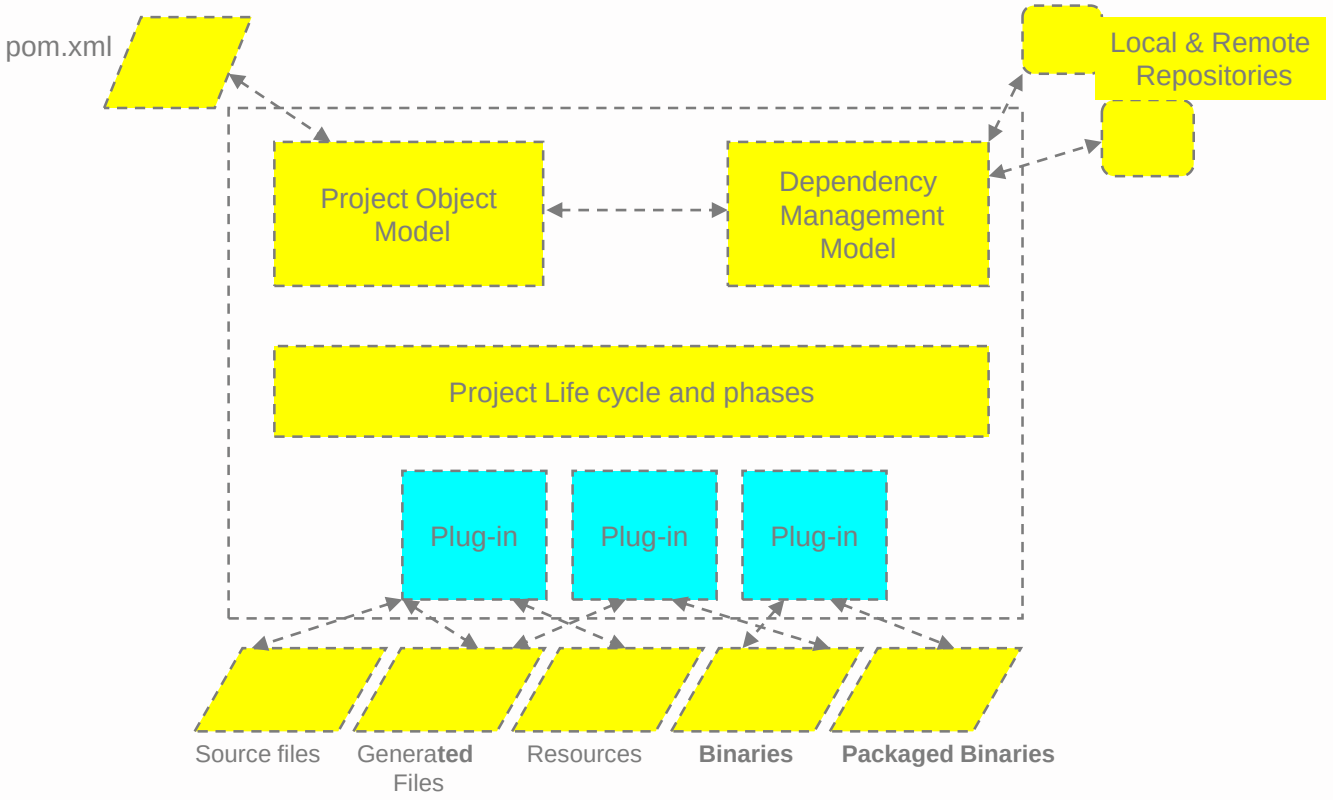
Apache Maven

- Conventions based ,know where source code is present
- Maven's Compiler plugin put the bytecode in target/classes, and it produces a JAR file in target.
- Maven is declarative with the use of a **pom.xml**

Maven Repositories

- Maven repositories store a set of artifacts which are used by Maven during dependency resolution for a project.
- Local repositories can be accessed on the local hard disk.
- Remote repositories can be accessed through the network.
- An artifact is bundled as a JAR file which contains the binary library or executable.
- An artifact can also be a war or an ear.
- After downloading repository, Maven will always look for the artifact in the local repository before looking elsewhere.

Physical Overview of Maven 2



POM File

- The src/test/java directory contains he pom.xml is the project's Project Object Model, or POM.
- POM is Maven’s understanding of a project.
- Dependencies are specified as a part of pom.xml file.

- Dependent components are known as artifacts, they are resolved in remote repositories are download to the local repository.

- The plug-ins are handled as artifacts by the dependency management model and are downloaded on demand.

POM.XML

```
- - - - schema definations ---
<groupId>com.mycompany.app</groupId>
<artifactId>my-app</artifactId>
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<name>Maven Quick Start Archetype</name>
<url>http://maven.apache.org</url>
<dependencies>
<dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
</dependency>
</dependencies>
</project>
```

POM.XML

- **project**
  - This is the top-level element in all Maven pom.xml files.
- **groupId**
  - This element indicates the unique identifier of the organization or group that created the project. The groupId is one of the key identifiers of a project and is typically based on the fully qualified domain name of your organization.
- **artifactId**
  - This element indicates the unique base name of the primary artifact being generated by this project. The primary artifact for a project is typically a JAR file.

POM.XML

- packaging
  - This element indicates the package type to be used by this artifact (e.g. JAR, WAR, EAR, etc.).
- Executing Phases of Application
- mvn compile
  - To Compile the application
- Mvn test
  - To Test the application

Phases in Maven

- *A phase is a step in the build lifecycle, which* is an ordered sequence of phases. When a phase is given, Maven will execute every phase in the sequence up to and including the one defined.

- compile:
  - compile the source code of the project
- package:
  - take the compiled code and package it in its distributable format, such as a JAR.
- deploy:
  - done in an integration or release environment, copies the final package to the remote

Jenkins

Introduction

- Jenkins is a CI/CD tool
  - Can be used for automation of common tasks like building, testing and deploying applications
- "Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible." -- Martin Fowler
- Continuous Integration:
  - The Goal: Improve quality, Decrease delivery time
  - Integrate and test code changes early and often
  - Automate the Build and Deployment
  - Make results visible to all

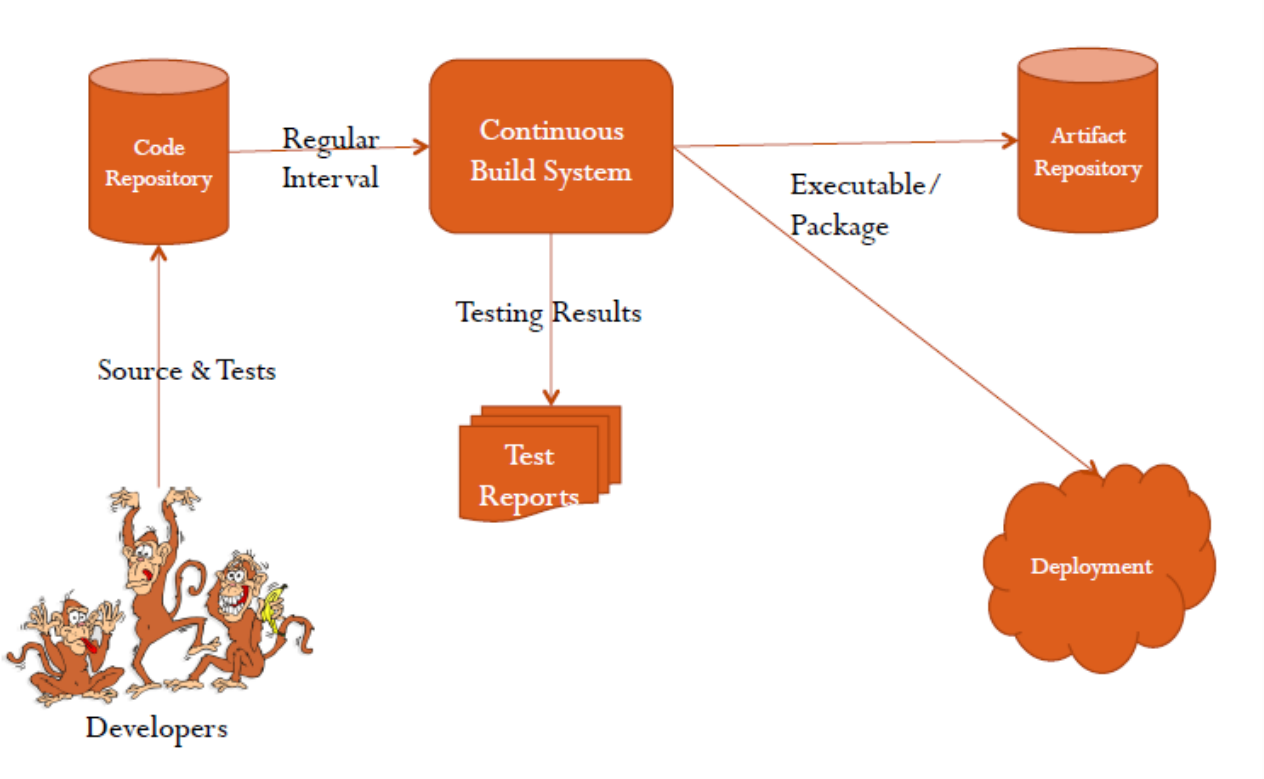
CI

- At a regular frequency (ideally at every commit), the system is:
  - Integrated
    - All changes up until that point are combined into the project
  - Built
    - The code is compiled into an executable or packaged
  - Tested
    - Automated test suites are run
  - Archived
    - Versioned and stored so it can be distributed as is, if desired
  - Deployed
    - Loaded onto a system where the developers can interact with it

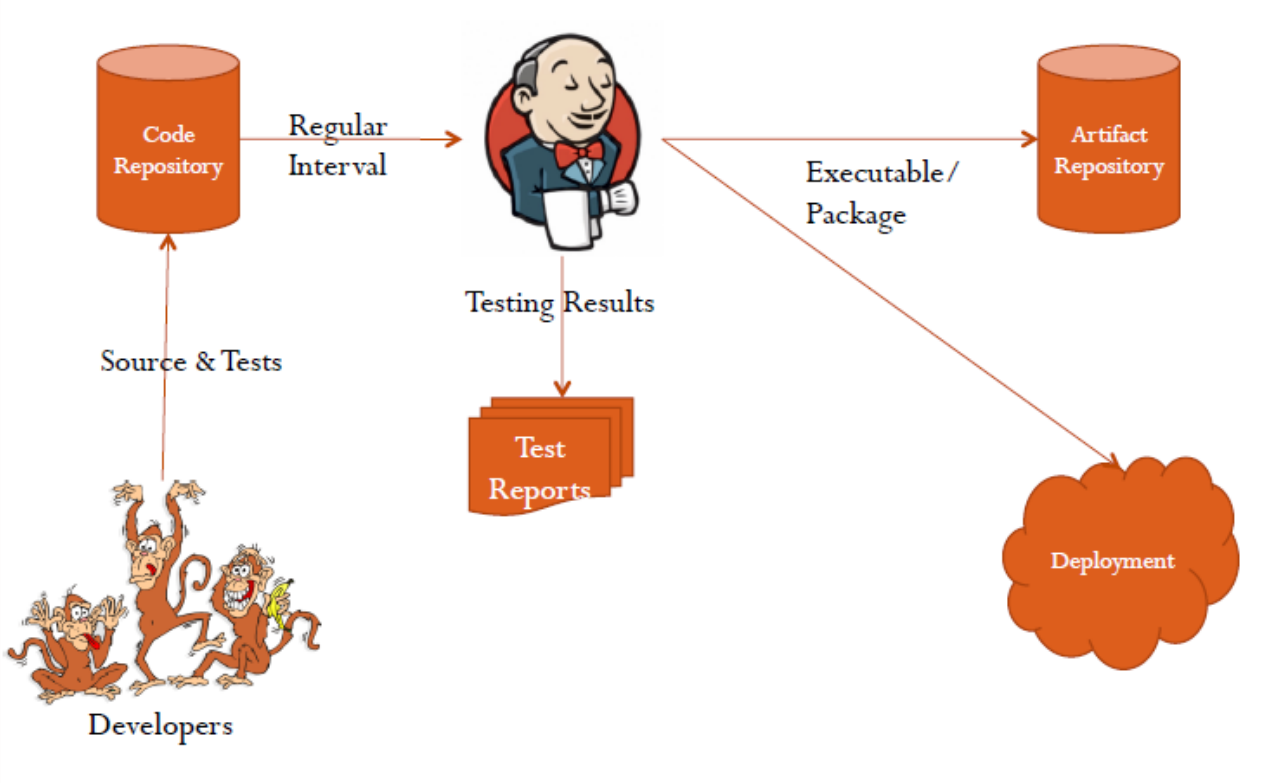
CI Benefits

- Immediate bug detection
- No integration step in the lifecycle
- A deployable system at any given point
- Record of evolution of the project

CI Workflow



Role of Jenkins



Workflow

- Checkout code from SCM (SVN, GIT, etc)
- Make changes to code (bug fixes, new features)
- Write and Run automated tests
- Merge with latest changes from SCM
- Commit code
- Run a build on a clean machine (CI Server)

What can Jenkins do?

- Generate test reports
- Integrate with many different Version Control Systems
- Push to various artifact repositories
- Deploys directly to production or test environments
- Notify stakeholders of build status
- …and much more

Jenkins for a Developer

- Easy to install
  - Download one file – jenkins.war
  - Run one command – java –jar jenkins.war
- Easy to use
  - Create a new job – checkout and build a small project
  - Checkin a change – watch it build
  - Create a test – watch it build and run
  - Fix a test – checkin and watch it pass
- Multi-technology
  - Build C, Java, C#, Python, Perl, SQL, etc.
  - Test with Junit, Nunit, MSTest, etc.


LTI
Let's Solve